```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

```
_$

Sy
---
NT$
NT$
NT$
NT$
NT$
NT$


NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$

NT$

NT$
NT$
NT$
NT$
NT$
NT$
NT$


NT$
NT$
NT$
NT$
NT$
NT$
PI
```

```
**FILE**ID**RMODIRSCN                                  C 5









RRRRRRRR    MM      MM    000000    DDDDDDDD      IIIIII    RRRRRRRR      SSSSSSSS    CCCCCCCC  NN      NN
RRRRRRRR    MM      MM    000000    DDDDDDDD      IIIIII    RRRRRRRR      SSSSSSSS    CCCCCCCC  NN      NN
RR      RR  MMMM  MMMM  00      00  DD      DD      II      RR      RR  SS          CC          NN      NN
RR      RR  MMMM  MMMM  00      00  DD      DD      II      RR      RR  SS          CC          NN      NN
RR      RR  MM  MM  MM  00    0000  DD      DD      II      RR      RR  SS          CC          NNNN    NN
RR      RR  MM  MM  MM  00    0000  DD      DD      II      RR      RR  SS          CC          NNNN    NN
RRRRRRRR    MM      MM  00  00  00  DD      DD      II      RRRRRRRR      SSSSSS    CC          NN  NN  NN
RRRRRRRR    MM      MM  00  00  00  DD      DD      II      RRRRRRRR      SSSSSS    CC          NN  NN  NN
RR  RR      MM      MM  0000    00  DD      DD      II      RR  RR              SS  CC          NN    NNNN
RR  RR      MM      MM  0000    00  DD      DD      II      RR  RR              SS  CC          NN    NNNN
RR    RR    MM      MM  00      00  DD      DD      II      RR    RR            SS  CC          NN      NN  ....
RR    RR    MM      MM  00      00  DD      DD      II      RR    RR            SS  CC          NN      NN  ....
RR      RR  MM      MM    000000    DDDDDDDD      IIIIII    RR      RR  SSSSSSSS      CCCCCCCC  NN      NN  ....
RR      RR  MM      MM    000000    DDDDDDDD      IIIIII    RR      RR  SSSSSSSS      CCCCCCCC  NN      NN  ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
     (2)        115           DEFINITIONS
     (3)        136           RM$READDIR, READ DIRECTORY FILE INTO MEMORY
     (4)        362           RM$DIRSCAN, SEARCH FOR NEXT FILE IN DIRECTORY
     (5)        541           NEXT_RECORD, SUBROUTINE TO FIND NEXT RECORD
     (6)        581           MATCH_VERSION, CHECK IF VERSION ENTRY MATCHES
     (7)        648           PARSE_NAME, PARSE FILE NAME STRING
     (8)        709           CONSTRUCT_NAME, CONSTRUCT RESULT FILE NAME STRING
     (9)        770           RETURN_FID, RETURN FID TO FIB BUFFER
```

```
0000            $BEGIN  RMODIRSCN,000,RM$RMSFILENAME,<READ DIRECTORY FILES>,-
0000                                                            <PIC,NOWRT>
0000
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ;++
0000    ; FACILITY: RMS32
0000    ;
0000    ; ABSTRACT:
0000    ;       This module performs the basic scanning of a directory
0000    ;       file as an optimization to performing ACP QIO's for every
0000    ;       file name in a directory file.
0000    ;
0000    ; ENVIRONMENT:
0000    ;       VAX/VMS
0000    ;
0000    ; AUTHOR:
0000    ;       Tim Halvorsen   October, 1979
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-012 JWT0171         Jim Teague              23-Mar-1984
0000    ;               Fix two broken branches.
0000    ;
0000    ;       V03-011 JWT0166         Jim Teague              20-Mar-1984
0000    ;               Use dynamically-allocated scratch page for storing
0000    ;               ATRs for QIOs.
0000    ;
0000    ;       V03-010 DGB0009         Donald G. Blair         01-Mar-1984
0000    ;               Change the way we call the ACP as part of the
0000    ;               restructuring necessary to implement access mode
0000    ;               protected files.  Also change to use RM$RMSFILENAME
0000    ;               psect.
0000    ;
0000    ;       V03-009 SRB0111         Steve Beckhardt         10-Feb-1984
0000    ;               Added support for cluster operation of directory cache.
```

```
0000  ;
0000  ;       V03-008 RAS0188         Ron Schaefer            11-Sep-1983
0000  ;               Properly initialize directory buffer BDB so that the
0000  ;               memory gets freed when the BDB is released.
0000  ;
0000  ;       V03-007 SHZ0002         Stephen H. Zalewski,    25-Jan-1983
0000  ;               Fix bug in SHZ0001 that did not rotate IFB$L_EBK_DISK to
0000  ;               IFB$L_EBK after doing a read from disk.
0000  ;
0000  ;       V03-006 SHZ0001         Stephen H. Zalewski,    16-Dec-1982  4:40
0000  ;               Keep swapped and unswapped longwords of EBK in different locations
0000  ;               in ifb.
0000  ;
0000  ;       V03-005 RAS0102         Ron Schaefer            4-Nov-1982
0000  ;               Correct broken branch from RAS0101.
0000  ;
0000  ;       V03-004 RAS0101         Ron Schaefer            22-Oct-1982
0000  ;               When the attempt to read in a directory buffer
0000  ;               failed, because the directory was too big, an ODS-I disk,
0000  ;               or because of random I/O errors; the BDB and buffer were
0000  ;               not released, causing a gradual consumption of P0 memory.
0000  ;
0000  ;       V03-003 KBT0204         Keith B. Thompson       23-Aug-1982
0000  ;               Reorganize psects
0000  ;
0000  ;       V03-002 DMW4001         DMWalp                  8-Jul-1982
0000  ;               Added check for zero size directory to be handled as
0000  ;               a bad directory, rather then causing a loop.
0000  ;
0000  ;       V03-001 TMK0001         Todd M. Katz            12-Jun-1982
0000  ;               Fix four broken branches by changing BSBWs (two to
0000  ;               RM$FCPFNC_ALT2 and one to RM$ALBDB and RM$FCPFNC each) to JSBs.
0000  ;
0000  ;       V02-070 JAK0070         J A Krycka      02-FEB-1982
0000  ;               Fix broken branch.
0000  ;
0000  ;       V02-009 JWH0002         Jeffrey W. Horn 02-DEC-1981
0000  ;               Add relative version numbers.  Also moved directory
0000  ;               BDB definitions to RMSINTSTR.MDL BDB definition.
0000  ;
0000  ;       V02-008 JWH0001         Jeffrey W. Horn 18-NOV-1981
0000  ;               Change multi-block logic to reflect removal of
0000  ;               DIR$V_PREVREC flag.
0000  ;
0000  ;       V02-007 JWH38296        Jeffrey W. Horn 22-SEP-1981
0000  ;               Round up EBK,FFB pair to nearest block boundry if not on
0000  ;               a block boundry.
0000  ;
0000  ;       V02-006 JAK0060         J A Krycka      01-JUL-1981
0000  ;               Fix broken branch.
0000  ;
0000  ;       V02-005 REFORMAT        Frederick E. Deen, Jr.          23-Jul-1980
0000  ;               This code was reformatted to adhere to RMS standards
0000  ;--
0000  ;
0000
```

```
                           0000    115          .SBTTL  DEFINITIONS
                           0000    116
                           0000    117  ;
                           0000    118  ;        Symbol definitions
                           0000    119  ;
                           0000    120
                           0000    121          $SSDEF
                           0000    122          $IFBDEF                                  ; IFAB definitions
                           0000    123          $FWADEF                                  ; FWA definitions
                           0000    124          $BDBDEF                                  ; BDB definitions
                           0000    125          $FIBDEF                                  ; FIB definitions
                           0000    126          $DEVDEF                                  ; Device Characteristics bits
                           0000    127          $CCBDEF                                  ; CCB definitions
                           0000    128          $UCBDEF                                  ; UCB definitions
                           0000    129          $FH2DEF                                  ; ODS-2 File header definitions
                           0000    130          $DIRDEF                                  ; Directory File definitions
                 00000007  0000    131          DIR$M_TYPE      = <1@DIR$S_TYPE-1>@DIR$V_TYPE
                           0000    132          $ATRDEF                                  ; Attribute List definitions
                           0000    133          $IODEF                                   ; I/O function codes
                           0000    134
```

```
                     0000          .SBTTL  RM$READDIR, READ DIRECTORY FILE INTO MEMORY
                     0000
                     0000  ;++
                     0000  ;
                     0000  ; RM$READDIR - Read directory file into memory
                     0000  ;
                     0000  ;       This routine attempts to read the directory file into
                     0000  ;       virtual memory owned by EXEC mode.  A directory buffer
                     0000  ;       descriptor is setup to contain the current status of
                     0000  ;       the search thru the directory file.
                     0000  ;
                     0000  ; INPUTS:
                     0000  ;
                     0000  ;       R11 = address of IMPURE AREA
                     0000  ;       R10 = FWA address
                     0000  ;       R9 = IFAB address
                     0000  ;       DID of FWA is used as FID of directory, channel is assigned.
                     0000  ;
                     0000  ; OUTPUTS:
                     0000  ;
                     0000  ;       R0 = status code (true if ok, error if could not be read)
                     0000  ;       FWA$L_DBD = address of DBD, 0 if none allocated
                     0000  ;       R1-R7 destroyed.
                     0000  ;--
                     0000
                     0000  RM$READDIR::
          04    E1   0000          BBC     #DEV$V_SDI,-                    ; branch if direc. structured
       06 69         0002                  IFB$L_PRIM_DEV(R9),10$
50   0828 8F    3C   0004          MOVZWL  #SS$_BADIRECTORY,R0             ; set unable to read directory
                05   0009  5$:     RSB
                     000A
                     000A  ;
                     000A  ;       Allocate a BDB (buffer descriptor block).  Miscellaneous
                     000A  ;       fields in the BDB will be used to hold scan context.
                     000A  ;
                     000A
          5A    DD   000A  10$:    PUSHL   R10                             ; save R10
     5A   59    D0   000C          MOVL    R9,R10                          ; RM$ALBDB wants IFB addr in R10
00000000'EF     16   000F          JSB     RM$ALBDB                        ; allocate BDB
          5A  8ED0   0015          POPL    R10                             ; restore R10
       EE 50    E9   0018          BLBC    R0,5$                           ; branch if error
     57   51    D0   001B          MOVL    R1,R7                           ; save BDB address
                     001E
                     001E  ;
                     001E  ;       Allocate  an ATR work area
                     001E  ;
          0E    BB   001E          PUSHR   #^M<R1,R2,R3>                   ; Otherwise, save regs
00000000'EF     16   0020          JSB     RM$GET1PAG                      ;  and get a scratch page
    58 AA 53    D0   0026          MOVL    R3,FWA$L_ATR_WORK(R10)          ;  for an ATR work area
       55 53    D0   002A          MOVL    R3,R5                           ; address of work area in R5
          0E    BA   002D          POPR    #^M<R1,R2,R3>                   ; Restore regs
                     002F
                     002F  ;
                     002F  ;       Setup attributes list
                     002F  ;
                     002F
          16    B0   002F          MOVW    #IFB$C_FHAEND-IFB$B_RFMORG,-    ; file attributes
```

```
                 85   0031              (R5)+
           85    04 B0 0032     MOVW    #ATR$C_RECATTR,(R5)+
     85     50 A9  9E 0035     MOVAB   IFB$B_RFMORG(R9),(R5)+
           85    04 B0 0039     MOVW    #4,(R5)+                       ; file characteristics longword
           85    03 B0 003C     MOVW    #ATR$C_UCHAR,(R5)+
     85        44 AA 9E 003F     MOVAB   FWA$W_UCHAR(R10),(R5)+
                      0043     ASSUME  FH2$B_STRUCLEV LT 8            ; assume STRUCLEV in 1st quadword
           85    08 B0 0043     MOVW    #8,(R5)+                       ; 1st 8 bytes of file hder blk
           85    0A B0 0046     MOVW    #ATR$C_HEADER,(R5)+
     85     01A8 CA 9E 0049     MOVAB   FWA$T_STATBLK(R10),(R5)+
                 65 D4 004E     CLRL    (R5)
                      0050
                      0050   ;
                      0050   ;  Setup FIB fields
                      0050   ;
                      0050
  10 AA     40 8F   9A 0050     MOVZBL  #FIB$C_LENGTH,FWA$Q_FIB(R10)   ; set length of FIB
     54     01F4 CA 9E 0055     MOVAB   FWA$T_FIBBUF(R10),R4           ; address of FIB
                 64 D4 005A     CLRL    FIB$L_ACCTL(R4)                ; allow other readers/writers
  04 A4     0A A4   D0 005C     MOVL    FIB$W_DID(R4),FIB$W_FID(R4)    ; copy DID to FID
  08 A4     0E A4   B0 0061     MOVW    FIB$W_DID+4(R4),FIB$W_FID+4(R4)
            0A A4   DD 0066     PUSHL   FIB$W_DID(R4)                  ; save did
     7E     0E A4   3C 0069     MOVZWL  FIB$W_DID+4(R4),-(SP)
            0A A4   D4 006D     CLRL    FIB$W_DID(R4)                  ; make sure did is 0 for
            0E A4   B4 0070     CLRW    FIB$W_DID+4(R4)                ;   acp operation.
                      0073
                      0073   ;
                      0073   ;  Request file attributes from the ACP
                      0073   ;
                      0073
                 00 DD 0073     PUSHL   #0                             ; P6 = 0
              58 AA DD 0075     PUSHL   FWA$L_ATR_WORK(R10)            ; P5 = addr of attribute list
                 7E 7C 0078     CLRQ    -(SP)                          ; P3/P4 = 0
                 00 DD 007A     PUSHL   #0                             ; P2 = 0
  50    0072 8F     3C 007C     MOVZWL  #IO$_ACCESS!IO$M_ACCESS,R0     ; ACP function code
   00000000'EF      16 0081     JSB     RM$FCPFNC                      ; call ACP and wait for reply
          56     50 D0 0087     MOVL    R0,R6                          ; save status
  54     01F4 CA    9E 008A     MOVAB   FWA$T_FIBBUF(R10),R4           ; r4 = address of fib
     0E A4       8E B0 008F     MOVW    (SP)+,FIB$W_DID+4(R4)          ; restore 3rd word of did
                 8E B5 0093     TSTW    (SP)+                          ; add 2 to SP
          0A A4 8ED0   0095     POPL    FIB$W_DID(R4)                  ; restore lower 2 words of did
                      0099   ;
                      0099   ;  Deallocate ATR work area
                      0099   ;
                 3E BB 0099     PUSHR   #^M<R1,R2,R3,R4,R5>            ; Yes, so save regs
     54       58 AA D0 009B     MOVL    FWA$L_ATR_WORK(R10),R4         ; Provide address of page
   00000000'EF      16 009F     JSB     RM$RET1PAG                     ;  to be deallocated
              58 AA D4 00A5     CLRL    FWA$L_ATR_WORK(R10)            ; Indicate no work area now
                 3E BA 00A8     POPR    #^M<R1,R2,R3,R4,R5>            ; Restore regs
                      00AA
          50     56 D0 00AA     MOVL    R6,R0                          ; restore status
              18 50 E8 00AD     BLBS    R0,20$                         ; branch if ok
                      00B0
                      00B0   ;
                      00B0   ;  Process error - deallocate BDB and return with status
                      00B0   ;
                      00B0
```

```
             0401 8F   BB  00B0   250 70$:   PUSHR   #^M<R0,R10>                         ; save status and FWA addr
                00B9   30  00B4   251        BSBW    DEACCESS                            ; deaccess file
             5A   59   D0  00B7   252        MOVL    R9,R10                              ; RM$RETBDB wants IFB addr in R10
             54   57   D0  00BA   253        MOVL    R7,R4                               ; BDB addr to R4
          00000000'EF  16  00BD   254        JSB     RM$RETBDB                           ; deallocate dir. buffer and BDB
             0401 8F   BA  00C3   255        POPR    #^M<R0,R10>                         ; restore STATUS and FWA addr
                       05  00C7   256 90$:   RSB                                         ; return with STATUS
                           00C8   257
                           00C8   258 ;
                           00C8   259 ;      If the directory is not a directory file or ODS-1
                           00C8   260 ;      structure, then exit with error.  We only read ODS-2.
                           00C8   261 ;
                           00C8   262
       50    0828 8F   3C  00C8   263 20$:   MOVZWL  #SS$_BADIRECTORY,R0                 ; assume invalid directory
       02    01AF CA   91  00CD   264        CMPB    FWA$T_STATBLK+FH2$B_STRUCLEV(R10),#2
                           00D2   265                                                    ; if not ODS-2
                  DC   12  00D2   266        BNEQ    70$                                 ; then exit with error
                  0D   E1  00D4   267        BBC     #FH2$V_DIRECTORY,-                  ; branch if not directory
            D7 44 AA       00D6   268                FWA$W_UCHAR(R10),70$
                           00D9   269
                           00D9   270 ;
                           00D9   271 ;      Allocate enough space to hold the entire directory file.
                           00D9   272 ;
                           00D9   273
    52    58 A9   10   9C  00D9   274        ROTL    #16,IFB$L_EBK_DISK(R9),R2           ; swap words of FILES-11 EBK
             5C A9     B5  00DE   275        TSTW    IFB$W_FFB(R9)                       ; First free byte at blk bound?
                  05   13  00E1   276        BEQL    25$                                 ; Branch if yes.
             5C A9     B4  00E3   277        CLRW    IFB$W_FFB(R9)                       ; No; Round up.
                  52   D6  00E6   278        INCL    R2                                  ; Increment block number
                  52   D7  00E8   279 25$:   DECL    R2                                  ; EOF file always points to last+1
                  C4   13  00EA   280        BEQL    70$                                 ; Test for zero size directory
          74 A9   52   D0  00EC   281        MOVL    R2,IFB$L_EBK(R9)                    ; set to nbr of blocks in file
          52  52  09   78  00F0   282        ASHL    #9,R2,R2                            ; convert blocks to bytes
51   52  0000FFFF 8F   CB  00F4   283        BICL3   #^XFFFF,R2,R1                       ; check if over 64k bytes
                  B2   12  00FC   284        BNEQ    70$                                 ; if over max. transfer size, skip i
          00000000'EF  16  00FE   285        JSB     RM$GETPAG                           ; allocate # blocks needed
               A9 50   E9  0104   286 170$:  BLBC    R0,70$                              ; branch if not available
          16 A7   52   B0  0107   287        MOVW    R2,BDB$W_SIZE(R7)                   ; store # bytes obtained
          2C A7   52   B0  010B   288        MOVW    R2,BDB$W_ALLOC_SIZE(R7)             ; store # bytes obtained
          18 A7   53   D0  010F   289        MOVL    R3,BDB$L_ADDR(R7)                   ; and address of buffer
          28 A7   53   D0  0113   290        MOVL    R3,BDB$L_ALLOC_ADDR(R7)             ; and address of buffer
                           0117   291
                           0117   292 ;
                           0117   293 ;      Remember DIRSEQ at the start of the transfer.  We will
                           0117   294 ;      check if afterwards to verify the validity of the file.
                           0117   295 ;
                           0117   296
          00000000'EF  16  0117   297        JSB     RM$GETCCB                           ; get CCB address in R1
             56   61   D0  011D   298        MOVL    CCB$L_UCB(R1),R6                    ; get UCB address
             00AC C6   B0  0120   299 30$:   MOVW    UCB$W_DIRSEQ(R6),-
                14 A7      0124   300                BDB$W_DIRSEQ(R7)                    ; save DIRSEQ value
                  13   19  0126   301        BLSS    35$                                 ; branch if AST is armed
                  57   DD  0128   302        PUSHL   R7                                  ; rearrange registers
       57    00AC C6   3E  012A   303        MOVAW   UCB$W_DIRSEQ(R6),R7                 ; r7 points to seq. # in UCB
          00000000'EF  16  012F   304        JSB     RM$ARM_DIRCACHE                     ; arm it
                  57 8ED0  0135   305        POPL    R7                                  ; restore r7
               E5 50   E8  0138   306        BLBS    R0,30$                              ; repeat, if successful
```

```
                      013B
                      013B  ;
                      013B  ;       Read the entire directory file into the buffer
                      013B  ;
                      013B
           7E    7C   013B  35$:    CLRQ    -(SP)                           ; P5/P6 = 0
       7E  01    7D   013D          MOVQ    #1,-(SP)                        ; P4 = 0, P3 = VBN 1
    7E  16 A7    3C   0140          MOVZWL  BDB$W_SIZE(R7),-(SP)            ; P2 = transfer size
        18 A7    DD   0144          PUSHL   BDB$L_ADDR(R7)                  ; P1 = buffer address
       50  31    3C   0147          MOVZWL  #IO$_READVBLK,R0                ; set I/O function code
  00000000'EF    16   014A          JSB     RM$FCPFNC_NOFIB                 ; read directory file
        B1 50    E9   0150          BLBC    R0,170$                         ; branch if error
                      0153
                      0153  ;
                      0153  ;       Check if DIRSEQ has changed while we were reading the
                      0153  ;       directory file.  If so, then keep trying until it goes
                      0153  ;       unchanged over the I/O.  This is to prevent invalid data
                      0153  ;       while in the middle of an ACP directory update.
                      0153  ;
                      0153
      00AC C6    B1   0153  40$:    CMPW    UCB$W_DIRSEQ(R6),-              ; DIRSEQ changed?
        14 A7         0157                  BDB$W_DIRSEQ(R7)
           C5    12   0159          BNEQ    30$                             ; if so, repeat transfer
                      015B
                      015B  ;
                      015B  ;       Deaccess the file
                      015B  ;
                      015B
           13    10   015B          BSBB    DEACCESS                        ; deaccess the file
                      015D
                      015D  ;
                      015D  ;       Initialize directory scan context
                      015D  ;
                      015D
     1C A7   01  D0   015D          MOVL    #1,BDB$L_VBN(R7)                ; set current VBN being searched
        18 A7    D0   0161          MOVL    BDB$L_ADDR(R7),-                ; set address of next record
        4C A7         0164                  BDB$L_RECORD(R7)
        48 A7    D4   0166          CLRL    BDB$L_VERSION(R7)               ; set version uninitialized
        20 A7    D4   0169          CLRL    BDB$L_LAST(R7)                  ; set last rec adr uninitzd
                      016C
       50  01    D0   016C          MOVL    #1,R0                           ; exit with success
           05         016F          RSB
                      0170
                      0170  ;
                      0170  ;       Deaccess the file
                      0170  ;
                      0170
                      0170  DEACCESS:
           7E    7C   0170          CLRQ    -(SP)                           ; P5/P6 = 0
           7E    7C   0172          CLRQ    -(SP)                           ; P3/P4 = 0
           7E    7C   0174          CLRQ    -(SP)                           ; P1/P2 = 0
       50  34    3C   0176          MOVZWL  #IO$_DEACCESS,R0                ; ACP function code
  00000000'EF    16   0179          JSB     RM$FCPFNC_NOFIB                 ; call the ACP and wait for reply
           05         017F          RSB                                     ; return with status
```

```
                        0180         .SBTTL  RM$DIRSCAN, SEARCH FOR NEXT FILE IN DIRECTORY
                        0180
                        0180 ;++
                        0180 ;
                        0180 ; RM$DIRSCAN - Search for next file in directory
                        0180 ;
                        0180 ;       This routine returns the next file name given the
                        0180 ;       file name search string and the current search context.
                        0180 ;
                        0180 ; Inputs:
                        0180 ;
                        0180 ;       R2/R3 = descriptor of file name/type/version string
                        0180 ;       R7 = directory BDB address
                        0180 ;       R9 = IFAB address
                        0180 ;       R10 = FWA address
                        0180 ;
                        0180 ; Outputs:
                        0180 ;
                        0180 ;       R0 = status code
                        0180 ;       result name string stored in result buffer
                        0180 ;       IFB$L_RNS_LEN(R9) = length of result name string
                        0180 ;       BDB$L_VERSION = address of version entry
                        0180 ;       BDB$L_RECORD = address of current record
                        0180 ;       BDB$L_VBN = block number currently being scanned
                        0180 ;--
                        0180
                        0180 RM$DIRSCAN::
       7E   0930 8F  3C 0180         MOVZWL  #SS$_NOMOREFILES,-(SP)          ; preset error status
               18 A7  D1 0185         CMPL    BDB$[_ADDR(R7),-                ; is this the first search?
               4C A7     0188                 BDB$L_RECORD(R7)
                  0A  12 018A         BNEQ    2$                              ; branch if not
               48 A7  D5 018C         TSTL    BDB$L_VERSION(R7)               ; if first search, version=0
                  05  12 018F         BNEQ    2$                              ; branch if not first search
       6E   0910 8F  3C 0191         MOVZWL  #SS$_NOSUCHFILE,(SP)            ; if so, return NOSUCHFILE on error
                         0196
                         0196 ;
                         0196 ;       Save the input parameters on the stack as:
                         0196 ;               (SP) = quadword descriptor of file name and type
                         0196 ;               4(SP) = version number (binary), -1=all, 0=highest
                         0196 ;
                         0196
                0133  30 0196 2$:     BSBW    PARSE_NAME                      ; parse into string and version #
               04 50  E8 0199         BLBS    R0,5$                           ; branch if successful
            5E   04  C0 019C         ADDL    #4,SP                           ; POP status longword
                  05     019F         RSB                                     ; return with status from PARSE_NAME
                  1C  BB 01A0 5$:     PUSHR   #^M<R2,R3,R4>                   ; save parameters on STACK
                         01A2
                         01A2 ;
                         01A2 ;       Loop over each block of the directory until EOF
                         01A2 ;
                         01A2
               1C A7  D1 01A2 10$:    CMPL    BDB$L_VBN(R7),-                 ; EOF yet?
               74 A9     01A5                 IFB$L_EBK(R9)
                  03  1B 01A7         BLEQU   15$                             ; if not, continue
                008A  31 01A9         BRW     600$                            ; otherwise, exit with failure
    51    1C A7  01  C3 01AC 15$:    SUBL3   #1,BDB$L_VBN(R7),R1             ; block number - 1
    51  00000200 8F  C4 01B1         MULL    #512,R1                         ; compute offset into directory
```

```
          51   18 A7    C0  01B8   419          ADDL    BDB$L_ADDR(R7),R1                ; address of current block
         55  01FF C1    9E  01BC   420          MOVAB   512-1(R1),R5                     ; R5 = last byte of the block
          56   4C A7    D0  01C1   421          MOVL    BDB$L_RECORD(R7),R6              ; R6 = address of current record
                            01C5   422
                            01C5   423 ;
                            01C5   424 ;        Loop over each record within the block
                            01C5   425 ;
                            01C5   426
        FFFF 8F   66    B1  01C5   427 20$:     CMPW    DIR$W_SIZE(R6),#^XFFFF           ; end of block marker?
                  76    13  01CA   428          BEQL    300$                             ; if so, skip to next block
                            01CC   429
                            01CC   430 ;
                            01CC   431 ;        Verify the length field of the current record
                            01CC   432 ;
                            01CC   433
             50   66    3C  01CC   434          MOVZWL  DIR$W_SIZE(R6),R0                ; get length of record
         53  02 A046    9E  01CF   435          MOVAB   2(R0)[R6],R3                     ; R3 = address of next record
             55   53    D1  01D4   436          CMPL    R3,R5                            ; outside of block boundary?
                  03    1B  01D7   437          BLEQU   25$                              ; branch if ok
                0089    31  01D9   438          BRW     ERRDIR                           ; exit with illegal dir format
                            01DC   439
                            01DC   440 ;
                            01DC   441 ;        Pick up address of current version.  If none, start at first.
                            01DC   442 ;
                            01DC   443
          52   05 A6    9A  01DC   444 25$:     MOVZBL  DIR$B_NAMECOUNT(R6),R2           ; R2 = # chars in name string
          54   48 A7    D0  01E0   445          MOVL    BDB$L_VERSION(R7),R4             ; R4 = address of current version
                  27    12  01E4   446          BNEQ    50$                              ; branch if version ok
         54  07 A642    9E  01E6   447          MOVAB   DIR$C_LENGTH+1(R6)[R2],R4        ; skip to first version
         54  54   01    CB  01EB   448          BICL3   #1,R4,R4
          50   F8 A5    9E  01EF   449          MOVAB   -DIR$C_VERSION(R5),R0            ; highest allowable within block
             50   54    D1  01F3   450          CMPL    R4,R0                            ; outside of block boundary?
                  6D    1A  01F6   451          BGTRU   ERRDIR                           ; yes, bad directory format
                            01F8   452
                            01F8   453
                            01F8   454 ;
                            01F8   455 ;        First version in record, see if we are starting a new file name
                            01F8   456 ;
                            01F8   457
                  3C    BB  01F8   458 30$:     PUSHR   #^M<R2,R3,R4,R5>
          51   20 A7    D0  01FA   459          MOVL    BDB$L_LAST(R7),R1
                  08    13  01FE   460          BEQL    40$                              ; branch if there wasn't one
  06 A6   06 A1   52    29  0200   461          CMPC3   R2,DIR$T_NAME(R1),DIR$T_NAME(R6); same as last record?
                  03    13  0206   462          BEQL    45$                              ; branch if so
               24 A7    D4  0208   463 40$:     CLRL    BDB$L_VERCOUNT(R7)               ; first version of this filename
                  3C    BA  020B   464 45$:     POPR    #^M<R2,R3,R4,R5>                 ; restore registers
                            020D   465
                            020D   466 ;
                            020D   467 ;        Determine if this record matches the input name and type
                            020D   468 ;
                            020D   469
                  3C    BB  020D   470 50$:     PUSHR   #^M<R2,R3,R4,R5>
          53   06 A6    9E  020F   471          MOVAB   DIR$T_NAME(R6),R3                ; R2/R3 = name being checked
          54   10 AE    7D  0213   472          MOVQ    4*4(SP),R4                       ; R4/R5 = name pattern
         00000000'EF    16  0217   473          JSB     FMG$MATCH_NAME                   ; check if name matches
                  3C    BA  021D   474          POPR    #^M<R2,R3,R4,R5>                 ; restore registers
               16 50    E9  021F   475          BLBC    R0,200$                          ; if not, exit this record
```

```
                        0222   476
                        0222   477
                        0222   478
                        0222   479 ;
                        0222   480 ;        Loop through each version entry looking for the desired one.
                        0222   481 ;
                        0222   482
         53    54  D1   0222   483 60$:     CMPL    R4,R3                          ; past last version?
               11  1E   0225   484          BGEQU   200$                           ; branch if so
    50      08 AE  D0   0227   485          MOVL    8(SP),R0                       ; pick up desired version #
             0070  30   022B   486          BSBW    MATCH_VERSION                  ; check if version entry matches
            1F 50  E8   022E   487          BLBS    R0,500$                        ; branch if match found
         54    08  C0   0231   488          ADDL    #DIR$C_VERSION,R4              ; skip to next version entry
               EC  11   0234   489          BRB     60$                            ; and keep looking
                        0236   490
                        0236   491 ;
                        0236   492 ;        No match found.  Exit with failure
                        0236   493 ;
                        0236   494
               33  11   0236   495 600$:    BRB     EXIT                           ; exit with status
                        0238   496
                        0238   497 ;
                        0238   498 ;        Skip to next record in the file
                        0238   499 ;
                        0238   500
               38  10   0238   501 200$:    BSBB    NEXT_RECORD                    ; skip to next record
            28 50  E9   023A   502          BLBC    R0,ERRDIR                      ; if error, bad format directory
            48 A7  D4   023D   503          CLRL    BDB$L_VERSION(R7)              ; mark no version address yet
               83  11   0240   504          BRB     20$                            ; scan this new record
                        0242   505
                        0242   506 ;
                        0242   507 ;        We have searched an entire block.  Skip to the next block.
                        0242   508 ;
                        0242   509
            1C A7  D6   0242   510 300$:    INCL    BDB$L_VBN(R7)                  ; increment block number
   4C A7    01 A5  9E   0245   511          MOVAB   1(R5),BDB$L_RECORD(R7)         ; set record address to next block
            48 A7  D4   024A   512          CLRL    BDB$L_VERSION(R7)              ; clear version address
             FF52  31   024D   513          BRW     10$                            ; search this new block
                        0250   514
                        0250   515 ;
                        0250   516 ;        Match found.  Exit with success
                        0250   517 ;
                        0250   518
   4C A7       56  D0   0250   519 500$:    MOVL    R6,BDB$L_RECORD(R7)            ; save record address
            08 A4  9E   0254   520          MOVAB   DIR$C_VERSION(R4),-            ; set version address
            48 A7       0257   521                  BDB$L_VERSION(R7)              ;  to the next version entry
             0116  30   0259   522          BSBW    RETURN_FID                     ; return FID to FIB buffer
             00C3  30   025C   523          BSBW    CONSTRUCT_NAME                 ; construct result file name
    0C AE      01  D0   025F   524          MOVL    #1,12(SP)                      ; set return status = successful
               06  11   0263   525          BRB     EXIT
                        0265   526
                        0265   527 ;
                        0265   528 ;        Illegal directory file format
                        0265   529 ;
                        0265   530
0C AE    0828 8F  3C    0265   531 ERRDIR: MOVZWL  #SS$_BADIRECTORY,12(SP)         ; set error status
                        026B   532
```

```
                    026B   533 ;
                    026B   534 ;        Deallocate directory buffer
                    026B   535 ;
                    026B   536
     5E    0C   C0  026B   537 EXIT:    ADDL    #3*4,SP                         ; remove parameters from STACK
          50  8ED0  026E   538          POPL    R0                              ; get return status
                05  0271   539          RSB
```

```
                        0272            .SBTTL  NEXT_RECORD, SUBROUTINE TO FIND NEXT RECORD
                        0272
                        0272    ;++
                        0272    ;
                        0272    ; NEXT_RECORD - Find next record
                        0272    ;
                        0272    ;       This subroutine is called to skip to the next record
                        0272    ;       in the directory file.
                        0272    ;
                        0272    ; Inputs:
                        0272    ;
                        0272    ;       R6 = address of current record
                        0272    ;       R5 = address of last byte of current block
                        0272    ;
                        0272    ; Outputs:
                        0272    ;
                        0272    ;       R0 = true if successful, false if illegal directory format
                        0272    ;--
                        0272
                        0272    NEXT_RECORD:
          50    66  3C  0272            MOVZWL  DIR$W_SIZE(R6),R0               ; get length of current record
          0E    50  D1  0275            CMPL    R0,-                            ; minimum length allowable
                        0278                    #DIR$C_LENGTH+DIR$C_VERSION
                21  1F  0278            BLSSU   80$                             ; branch if illegal
    56    02 A046   9E  027A            MOVAB   2(R0)[R6],R6                    ; advance to next record in block
          55    56  D1  027F            CMPL    R6,R5                           ; check if exceeded block boundary
                17  1A  0282            BGTRU   80$                             ; branch if illegal format
  FFFF 8F       66  B1  0284            CMPW    DIR$W_SIZE(R6),#^XFFFF          ; if end-of-block marker, skip check
                0C  13  0289            BEQL    20$                             ; branch if so
       0D       56  E8  028B            BLBS    R6,80$                          ; all records must be word aligned
       0A       66  E8  028E            BLBS    DIR$W_SIZE(R6),80$              ; and the size must also be in words
                        0291            ASSUME  DIR$C_FID EQ 0
       04       A6  93  0291            BITB    DIR$B_FLAGS(R6),-               ; check if DIR$V_TYPE=DIR$C_FID
                    07  0294                    #DIR$M_TYPE
                04  12  0295            BNEQ    80$                             ; branch if not
          50    01  D0  0297    20$:    MOVL    #1,R0
                    05  029A            RSB
                50  D4  029B    80$:    CLRL    R0                              ; illegal format record
                    05  029D            RSB
```

```
                      029E          .SBTTL  MATCH_VERSION, CHECK IF VERSION ENTRY MATCHES
                      029E
                      029E  ;++
                      029E  ;
                      029E  ; MATCH_VERSION - Check if version entry matches
                      029E  ;
                      029E  ;       This routine checks if the current version entry matches
                      029E  ;       the requested version number.
                      029E  ;
                      029E  ; INPUTS:
                      029E  ;
                      029E  ;       R0 = requested version number
                      029E  ;       R7 = directory BDB address
                      029E  ;       R6 = address of current record
                      029E  ;       R4 = address of current version entry
                      029E  ;       R2 = DIR$B_NAMECOUNT(R6)
                      029E  ;
                      029E  ; OUTPUTS:
                      029E  ;
                      029E  ;       R0 = true if matches, else false
                      029E  ;       R1 destroyed.
                      029E  ;--
                      029E
                      029E  MATCH_VERSION:
        0B A7    95   029E          TSTB    BDB$B_VERTYP(R7)                ; check if wild version
           1E    14   02A1          BGTR    70$                             ; if all, match immediately
           50    D5   02A3          TSTL    R0                              ; check version number
           10    19   02A5          BLSS    60$                             ; branch if relative
           05    12   02A7          BNEQ    50$                             ; branch if specific value
                      02A9
                      02A9  ;
                      02A9  ;       Version = 0, match only highest version
                      02A9  ;
                      02A9
        24 A7    D5   02A9          TSTL    BDB$L_VERCOUNT(R7)              ; is it first vers of file
           13    13   02AC          BEQL    70$                             ; match if yes
                      02AE
                      02AE  ;
                      02AE  ;       Version = #, check if matches
                      02AE  ;
                      02AE
     64    50    B1   02AE  50$:    CMPW    R0,DIR$W_VERSION(R4)            ; check if matches
           0E    13   02B1          BEQL    70$                             ; if so, match immediately
           50    D4   02B3          CLRL    R0                              ; no match
           0D    11   02B5          BRB     EXIT_MATCH
                      02B7
                      02B7  ;
                      02B7  ;       Relative version number, see if this one
                      02B7  ;
                      02B7
  24 A7    50    D1   02B7  60$:    CMPL    R0,BDB$L_VERCOUNT(R7)           ; current relative offset?
           04    13   02BB          BEQL    70$                             ; yes, successful match
           50    D4   02BD          CLRL    R0                              ; no match
           03    11   02BF          BRB     EXIT_MATCH
                      02C1
                      02C1
                      02C1  ;
```

```
                     02C1   638 ;       Report successful match
                     02C1   639 ;
                     02C1   640
      50    01   D0  02C1   641 70$:    MOVL    #1,R0                           ; match
                     02C4   642
                     02C4   643 EXIT_MATCH:
         24 A7   D7  02C4   644         DECL    BDB$L_VERCOUNT(R7)              ; bump version count
   20 A7    56   D0  02C7   645         MOVL    R6,BDB$L_LAST(R7)               ; save adr of this record
                 05  02CB   646         RSB
```

```
                        02CC          .SBTTL  PARSE_NAME, PARSE FILE NAME STRING
                        02CC
                        02CC    ;++
                        02CC    ;
                        02CC    ; PARSE_NAME - Parse file name string
                        02CC    ;
                        02CC    ;
                        02CC    ;       This routine parses the file name string into a
                        02CC    ;       string composed of the file name and type and a
                        02CC    ;       binary number representing the version number.
                        02CC    ;       It is assumed that all portions of the file string
                        02CC    ;       are present (file name, type and version).
                        02CC    ;
                        02CC    ; INPUTS:
                        02CC    ;
                        02CC    ;       R2/R3 = string descriptor
                        02CC    ;       R7 = directory BDB address
                        02CC    ;
                        02CC    ; OUTPUTS:
                        02CC    ;
                        02CC    ;       R0 = status
                        02CC    ;       R2/R3 = descriptor of file name/type string
                        02CC    ;       R4 = version number
                        02CC    ;--
                        02CC
                        02CC    PARSE_NAME:
              54   D4   02CC            CLRL    R4                              ; preset result to 0
           0B A7   94   02CE            CLRB    BDB$B_VERTYP(R7)                ; preset version type to nonwild
        51    01   D0   02D1            MOVL    #1,R1                           ; preset base factor to 1
              52   D7   02D4    10$:    DECL    R2                              ; decrement string size
              32   15   02D6            BLEQ    40$                             ; get out if string runs out
      50    6342   9A   02D8            MOVZBL  (R3)[R2],R0                     ; get last character
        2A    50   91   02DC            CMPB    R0,#^A'*'                       ; asterisk means all versions
              15   13   02DF            BEQL    20$                             ; branch if so
        50    30   82   02E1            SUBB    #^A'0',R0                       ; check lower bounds
              16   19   02E4            BLSS    30$                             ; branch if not numeric
        09    50   91   02E6            CMPB    R0,#9                           ; check upper bounds
              11   1A   02E9            BGTRU   30$                             ; branch if not numeric
        50    51   C4   02EB            MULL    R1,R0                           ; multiply by base factor
        54    50   C0   02EE            ADDL    R0,R4                           ; add to result
        51    0A   C4   02F1            MULL    #10,R1                          ; multiply base by 10
              DE   11   02F4            BRB     10$
                        02F6
     0B A7    01   90   02F6    20$:    MOVB    #1,BDB$B_VERTYP(R7)             ; set version type to 1 (wild)
              D8   11   02FA            BRB     10$                             ; and continue to remove semicolon
                        02FC
      50    6342   9A   02FC    30$:    MOVZBL  (R3)[R2],R0                     ; get last character
        2D    50   91   0300            CMPB    R0,#^A'-'                       ; negative version?
              05   12   0303            BNEQ    40$                             ; no, branch
        54    54   CE   0305            MNEGL   R4,R4                           ; yes, negate binary version #
              52   D7   0308            DECL    R2                              ; decrement string size
      50    6342   9A   030A    40$:    MOVZBL  (R3)[R2],R0                     ; get last character
        3B    50   91   030E            CMPB    R0,#^A';'                       ; terminator must be ';'
              0B   13   0311            BEQL    90$                             ; if so, exit ok
        2E    50   91   0313            CMPB    R0,#^A'.'                       ; alternate syntax
              06   13   0316            BEQL    90$                             ; if so, exit ok
  50    0820  8F   3C   0318            MOVZWL  #SS$_BADFILEVER,R0              ; set error status
```

```
                       05  031D    705          RSB                                     ; exit
         50     01     D0  031E    706 90$:     MOVL    #1,R0                           ; success
                       05  0321    707          RSB
```

```
                          0322   709         .SBTTL  CONSTRUCT_NAME, CONSTRUCT RESULT FILE NAME STRING
                          0322   710
                          0322   711 ;++
                          0322   712 ;
                          0322   713 ; CONSTRUCT_NAME - Construct RESULT FILE NAME STRING
                          0322   714 ;
                          0322   715 ;
                          0322   716 ;       This routine constructs a result file name string
                          0322   717 ;       given the address of the record and version entry
                          0322   718 ;       which represents the file name.
                          0322   719 ;
                          0322   720 ; INPUTS:
                          0322   721 ;
                          0322   722 ;       R10 = FWA address
                          0322   723 ;       R9 = IFAB address
                          0322   724 ;       R6 = address of record
                          0322   725 ;       R4 = address of version entry
                          0322   726 ;
                          0322   727 ; OUTPUTS:
                          0322   728 ;
                          0322   729 ;       R0-R3,R5 destroyed.
                          0322   730 ;       Result string is copied to buffer FWA$Q_NAME
                          0322   731 ;       IFB$L_RNS_LEN = result string length
                          0322   732 ;--
                          0322   733
                          0322   734 CONSTRUCT_NAME:
     53   0174 CA    D0   0322   735         MOVL    FWA$Q_NAME+4(R10),R3            ; set result buffer address
       50   05 A6    9A   0327   736         MOVZBL  DIR$B_NAMECOUNT(R6),R0          ; file name size
                54   DD   032B   737         PUSHL   R4                              ; save version entry address
  63    06 A6   50   28   032D   738         MOVC    R0,DIR$T_NAME(R6),(R3)          ; copy to result buffer
        83      3B   90   0332   739         MOVB    #^A';',(R3)+                    ; append semi-colon
             54 8ED0      0335   740         POPL    R4                              ; restore version address
                          0338   741
                          0338   742 ;
                          0338   743 ;       Bypass leading zeros
                          0338   744 ;
                          0338   745
        50      64   3C   0338   746         MOVZWL  DIR$W_VERSION(R4),R0            ; get binary version number
  51    00002710 8F  D0   033B   747         MOVL    #10000,R1                       ; divisor (largest is 65000)
                0B   13   0342   748 10$:    BEQL    20$                             ; branch if divisor hits zero
  52    50      51   C7   0344   749         DIVL3   R1,R0,R2                        ; get upper digit
                05   12   0348   750         BNEQ    20$                             ; branch if non-zero digit
        51      0A   C6   034A   751         DIVL    #10,R1                          ; go to next position down
                F3   11   034D   752         BRB     10$                             ; continue skipping leading zeros
                          034F   753
                          034F   754 ;
                          034F   755 ;       Convert binary version number to ASCII
                          034F   756 ;
                          034F   757
                51   D5   034F   758 20$:    TSTL    R1                              ; check divisor
                17   13   0351   759 30$:    BEQL    50$                             ; branch if divisor hits zero
        52  50  51   C7   0353   760         DIVL3   R1,R0,R2                        ; get upper digit
        83  52  30   81   0357   761         ADDB3   #^A'0',R2,(R3)+                 ; write character to buffer
  7E    00  50  01   7A   035B   762         EMUL    #1,R0,#0,-(SP)                  ; convert version to quadword
  50    50  8E  51   7B   0360   763         EDIV    R1,(SP)+,R0,R0                  ; version = version mod divisor
            51  0A   C6   0365   764         DIVL    #10,R1                          ; go to next position down
                E7   11   0368   765         BRB     30$                             ; loop until done
```

```
      6C A9   53   0174 CA   C3  036A    766 50$:    SUBL3   FWA$Q_NAME+4(R10),R3,-         ; return result length
                                 0371    767                 IFB$L_RNS_LEN(R9)
                             05  0371    768         RSB                                    ; exit
```

```
RM
Sy

TP
TP
TP
TP
TP
TP
TP
TP
TP
TP
TP
TP
TP

PS
--
.
RM
$A
_R

Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As

Th
22
Th
27
15
```

```
                          0372              .SBTTL  RETURN_FID, RETURN FID TO FIB BUFFER
                          0372
                          0372      ;++
                          0372      ;
                          0372      ; RETURN_FID - Return FID to FIB buffer
                          0372      ;
                          0372      ;
                          0372      ;       Return the FID of the matched directory file name
                          0372      ;       to the FIB buffer in the FWA.
                          0372      ;
                          0372      ; INPUTS:
                          0372      ;
                          0372      ;       R10 = FWA address
                          0372      ;       R4 = address of version entry
                          0372      ;
                          0372      ; Outputs:
                          0372      ;
                          0372      ;       The FID is copied to the FIB.
                          0372      ;--
                          0372
                          0372      RETURN_FID:
                          0372
                          0372      ;
                          0372      ;       Return FID to FIB buffer
                          0372      ;
                          0372
      50  01F4 CA     9E  0372              MOVAB   FWA$T_FIBBUF(R10),R0            ; address of FIB
             02 A4    D0  0377              MOVL    DIR$W_FID_NUM(R4),-             ; copy num and seq
             04 A0        037A                      FIB$W_FID_NUM(R0)
             06 A4    B0  037C              MOVW    DIR$W_FID_RVN(R4),-             ; copy relative volume
             08 A0        037F                      FIB$W_FID_RVN(R0)
                          0381
                          0381      ;
                          0381      ;       If the relative volume number is 0, then substitute
                          0381      ;       the relative volume number of its parent directory.
                          0381      ;
                          0381
             08 A0    95  0381              TSTB    FIB$B_FID_RVN(R0)               ; check if low byte zero
                05    12  0384              BNEQ    60$                             ; branch if not zero
             0E A0    90  0386              MOVB    FIB$B_DID_RVN(R0),-             ; substitute parent RVN
             08 A0        0389                      FIB$B_FID_RVN(R0)
                      05  038B      60$:    RSB                                     ; exit
```

```
                                 038C   813          .END
```

```
$$.PSECT_EP         = 00000000            IFB$C_FHAEND        = 00000066
$$RMSTEST           = 0000001A            IFB$L_EBK           = 00000074
$$RMS_PBUGCHK       = 00000010            IFB$L_EBK_DISK      = 00000058
$$RMS_TBUGCHK       = 00000008            IFB$L_PRIM_DEV      = 00000000
$$RMS_UMODE         = 00000004            IFB$L_RNS_LEN       = 0000006C
ATR$C_HEADER        = 0000000A            IFB$W_FFB           = 0000005C
ATR$C_RECATTR       = 00000004            IO$M_ACCESS         = 00000040
ATR$C_UCHAR         = 00000003            IO$_ACCESS          = 00000032
BDB$B_VERTYP        = 0000000B            IO$_DEACCESS        = 00000034
BDB$L_ADDR          = 00000018            IO$_READVBLK        = 00000031
BDB$L_ALLOC_ADDR    = 00000028            MATCH_VERSION         0000029E R     01
BDB$L_LAST          = 00000020            NEXT_RECORD           00000272 R     01
BDB$L_RECORD        = 0000004C            PARSE_NAME            000002CC R     01
BDB$L_VBN           = 0000001C            RETURN_FID            00000372 R     01
BDB$L_VERCOUNT      = 00000024            RM$ALBDB              ********    X  01
BDB$L_VERSION       = 00000048            RM$ARM_DIRCACHE       ********    X  01
BDB$W_ALLOC_SIZE    = 0000002C            RM$DIRSCAN            00000180 RG    01
BDB$W_DIRSEQ        = 00000014            RM$FCPFNC             ********    X  01
BDB$W_SIZE          = 00000016            RM$FCPFNC_NOFIB       ********    X  01
CCB$L_UCB           = 00000000            RM$GET1PAG            ********    X  01
CONSTRUCT_NAME        00000322 R     01   RM$GETCCB             ********    X  01
DEACCESS              00000170 R     01   RM$GETPAG             ********    X  01
DEV$V_SDI           = 00000004            RM$READDIR            00000000 RG    01
DIR$B_FLAGS         = 00000004            RM$RET1PAG            ********    X  01
DIR$B_NAMECOUNT     = 00000005            RM$RETBDB             ********    X  01
DIR$C_FID           = 00000000            SS$_BADFILEVER      = 00000820
DIR$C_LENGTH        = 00000006            SS$_BADIRECTORY     = 00000828
DIR$C_VERSION       = 00000008            SS$_NOMOREFILES     = 00000930
DIR$M_TYPE          = 00000007            SS$_NOSUCHFILE      = 00000910
DIR$S_TYPE          = 00000003            UCB$W_DIRSEQ        = 000000AC
DIR$T_NAME          = 00000006
DIR$V_TYPE          = 00000000
DIR$W_FID_NUM       = 00000002
DIR$W_FID_RVN       = 00000006
DIR$W_SIZE          = 00000000
DIR$W_VERSION       = 00000000
ERRDIR                00000265 R     01
EXIT                  0000026B R     01
EXIT_MATCH            000002C4 R     01
FH2$B_STRUCLEV      = 00000007
FH2$V_DIRECTORY     = 0000000D
FIB$B_DID_RVN       = 0000000E
FIB$B_FID_RVN       = 00000008
FIB$C_LENGTH        = 00000040
FIB$L_ACCTL         = 00000000
FIB$W_DID           = 0000000A
FIB$W_FID           = 00000004
FIB$W_FID_NUM       = 00000004
FIB$W_FID_RVN       = 00000008
FMG$MATCH_NAME        ********    X  01
FWA$L_ATR_WORK      = 00000058
FWA$Q_FIB           = 00000010
FWA$Q_NAME          = 00000170
FWA$T_FIBBUF        = 000001F4
FWA$T_STATBLK       = 000001A8
FWA$W_UCHAR         = 00000044
IFB$B_RFMORG        = 00000050
```

```
                                                  +----------------+
                                                  ! Psect synopsis !
                                                  +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
RM$RMSFILENAME                  0000038C  (   908.)  01 (  1.)    PIC    USR    CON    REL    GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE
$ABS$                           00000000  (     0.)  02 (  2.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE

                                               +------------------------+
                                               ! Performance indicators !
                                               +------------------------+

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                     30    00:00:00.13    00:00:01.30
Command processing                120    00:00:00.70    00:00:04.90
Pass 1                            499    00:00:19.80    00:00:57.78
Symbol table sort                   0    00:00:03.42    00:00:07.17
Pass 2                            146    00:00:03.81    00:00:10.38
Symbol table output                10    00:00:00.14    00:00:00.21
Psect synopsis output               2    00:00:00.02    00:00:00.02
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals              809    00:00:28.02    00:01:21.77

The working set limit was 1950 pages.
112711 bytes (221 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2182 non-local and 40 local symbols.
813 source lines were read in Pass 1, producing 15 object records in Pass 2.
22 pages of virtual memory were used to define 21 macros.

                                              +--------------------------+
                                              ! Macro library statistics !
                                              +--------------------------+

Macro library name                            Macros defined
------------------                            --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                       4
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                       4
_$255$DUA28:[SYSLIB]STARLET.MLB;2                    9
TOTALS (all libraries)                              17

2282 GETS were required to define 17 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RMODIRSCN/OBJ=OBJ$:RMODIRSCN MSRC$:RMODIRSCN/UPDATE=(ENH$:RMODIRSCN)+EXECML$/LIB+LIB$:RMS/LIB
```

0318 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY